AMENDMENTS TO THE CLAIMS

Please cancel claims 4-5 and 29-31 without prejudice. Kindly amend claims 1, 6-7, 17 and 32 as shown in the following listing of claims. Please add new claims 33-59. The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims

1. (currently amended) An apparatus for killing an instruction loaded into an instruction queue of a microprocessor during a first clock cycle and output from a bottom entry of the instruction queue during a second clock cycle subsequent to the first clock cycle, the apparatus comprising:

   a kill signal, for conveying a value generated during a third clock cycle subsequent to the first clock cycle;

   a kill queue, coupled to said kill signal, for loading said kill signal value generated during said third clock cycle, and for outputting said kill signal value during the second clock cycle;

   a load signal, coupled to said kill queue, for indicating during the second clock cycle whether the instruction was loaded into the bottom entry of the instruction queue during the first clock cycle, wherein if said load signal is true, said third clock cycle and the second clock cycle are a same clock cycle; and

   a valid signal, coupled to said kill queue, generated during the second clock cycle for indicating whether the instruction is to be executed by the microprocessor, wherein said valid signal is false if said kill signal value output by said kill queue during the second clock cycle is true.

2. (original) The apparatus of claim 1, wherein said third clock cycle is a same clock cycle as the second clock cycle.

3. (original) The apparatus of claim 1, wherein said third clock cycle is a clock cycle prior to the second clock cycle.

4-5. (canceled)

6. (currently amended) The apparatus of ~~claim 4~~claim 1, wherein if said load signal is false, the second clock cycle is subsequent to said third clock cycle.

7. (currently amended) The apparatus of ~~claim 4~~claim 1, further comprising:

   logic, coupled to said kill queue, for generating said valid signal during the second clock cycle based on said load signal and said kill signal value output by said kill queue.

8. (original) The apparatus of claim 1, wherein said kill queue comprises:

a plurality of entries, for storing a plurality of values of said kill signal generated during a corresponding plurality of clock cycles.

9. (original) The apparatus of claim 8, wherein each of said plurality of kill queue entries comprises a load data input, coupled to receive said kill signal.

10. (original) The apparatus of claim 8, wherein each of said plurality of kill queue entries comprises a hold data input, coupled to receive a current value of said entry.

11. (original) The apparatus of claim 8, wherein each of said plurality of kill queue entries comprises a shift data input, coupled to receive one of said plurality of values of said kill signal from one of said plurality of entries above said each of said plurality of kill queue entries.

12. (original) The apparatus of claim 8, wherein the instruction queue comprises a plurality of entries for storing a plurality of instructions, wherein said plurality of kill queue entries store corresponding said kill signal values for said plurality of instructions stored in said plurality of instruction queue entries.

13. (original) The apparatus of claim 1, wherein the instruction comprises a variable length instruction.

14. (original) The apparatus of claim 13, wherein said variable length instruction comprises an x86 architecture instruction.

15. (original) The apparatus of claim 13, wherein the instruction is provided to the instruction queue during the first clock cycle by an instruction formatter, said instruction formatter determining a length of the instruction.

16. (original) The apparatus of claim 1, wherein the instruction is output from the bottom entry of the instruction queue during the second clock cycle to an instruction translator for translating the instruction into one or more microinstructions to be selectively executed by the microprocessor based on said valid signal.

17. (currently amended) A method for killing an instruction in a microprocessor, the method comprising:

loading an instruction into a first queue during a first clock cycle;

generating a kill signal during a second clock cycle subsequent to said first clock cycle;

loading a value of said kill signal into a second queue during said second clock cycle;

determining whether said value in the second queue is true during a third clock cycle in which said instruction is output from a bottom entry of said first queue;

generating a load signal for indicating during said third clock cycle whether said instruction was loaded into a bottom entry of said first queue during said

first clock cycle, wherein if said load signal is true, said third clock cycle and said second clock cycle are a same clock cycle; and

foregoing execution of said instruction if said value is true.

18. (original) The method of claim 17, wherein said third clock cycle is a same clock cycle as said second clock cycle.

19. (original) The method of claim 17, wherein said third clock cycle is clock cycle subsequent to said second clock cycle.

20. (original) The method of claim 17, further comprising:

formatting said instruction prior to said loading said instruction into said first queue.

21. (original) The method of claim 17, further comprising:

determining whether said instruction is shifted down in said first queue after said loading said instruction into said first queue; and

shifting down said value of said kill signal in said second queue after said loading a value of said kill signal into a second queue, if said instruction is shifted down in said first queue.

22. (original) The method of claim 17, further comprising:

predicting said instruction is a taken branch instruction, prior to said loading said instruction into said first queue;

detecting a misprediction of said branch instruction; and

said generating said kill signal during said second clock cycle in response to said detecting said misprediction.

23. (original) The method of claim 22, wherein a branch target address cache of the microprocessor performs said predicting said instruction is a taken branch instruction.

24. (original) The method of claim 22, wherein said misprediction of said branch instruction comprises a misprediction of a length of said branch instruction.

25. (original) The method of claim 22, wherein said misprediction of said branch instruction comprises a misprediction of an address of said branch instruction.

26. (original) The method of claim 22, wherein said misprediction of said branch instruction comprises said branch instruction being a non-branch instruction.

27. (original) The method of claim 17, further comprising:

branching the microprocessor based on a prediction that a branch instruction is taken, wherein said instruction is sequential to said branch instruction; and

said generating said kill signal during said second clock cycle after said branching the microprocessor.

28. (original) The method of claim 17, wherein said instruction sequentially follows a branch instruction predicted taken, the method further comprising:

said generating said kill signal during said second clock cycle in response to detecting said branch instruction is predicted taken.

29-31. (canceled)

32. (currently amended) ~~A computer data signal embodied in a transmission medium~~A program embodied on a computer readable medium, comprising:

computer-readable program code for providing an apparatus for killing an instruction loaded into an instruction queue of a microprocessor during a first clock cycle and output from a bottom entry of the instruction queue during a second clock cycle subsequent to the first clock cycle, said program code comprising:

first program code for providing a kill signal, for conveying a value generated during a third clock cycle subsequent to the first clock cycle;

second program code for providing a kill queue, coupled to said kill signal, for loading said kill signal value generated during said third clock cycle, and for outputting said kill signal value during the second clock cycle;

third program code for providing a load signal, coupled to said kill queue, for indicating during the second clock cycle whether the instruction was loaded into the bottom entry of the instruction queue during the first clock cycle, wherein if said load signal is true, said third clock cycle and the second clock cycle are a same clock cycle; and

~~third~~ fourth program code for providing a valid signal, coupled to said kill queue, generated during the second clock cycle for indicating whether the instruction is to be executed by the microprocessor, wherein said valid signal is false if said kill signal value output by said kill queue during the second clock cycle is true.

33. (new) An apparatus for killing an instruction loaded into an instruction queue of a microprocessor during a first clock cycle and output from a bottom entry of the instruction queue during a second clock cycle subsequent to the first clock cycle, the apparatus comprising:

a kill signal, for conveying a value generated during a third clock cycle subsequent to the first clock cycle;

a kill queue, coupled to said kill signal, for loading said kill signal value generated during said third clock cycle, and for outputting said kill signal value during the second clock cycle;

a load signal, coupled to said kill queue, for indicating during the second clock cycle whether the instruction was loaded into the bottom entry of the instruction queue during the first clock cycle, wherein if said load signal is false, the second clock cycle is subsequent to said third clock cycle; and

a valid signal, coupled to said kill queue, generated during the second clock cycle for indicating whether the instruction is to be executed by the microprocessor, wherein said valid signal is false if said kill signal value output by said kill queue during the second clock cycle is true.

34. (new) The apparatus of claim 33, wherein said third clock cycle is a same clock cycle as the second clock cycle.

35. (new) The apparatus of claim 33, wherein said third clock cycle is a clock cycle prior to the second clock cycle.

36. (new) The apparatus of claim 33, wherein if said load signal is true, said third clock cycle and the second clock cycle are a same clock cycle.

37. (new) The apparatus of claim 33, further comprising:

logic, coupled to said kill queue, for generating said valid signal during the second clock cycle based on said load signal and said kill signal value output by said kill queue.

38. (new) The apparatus of claim 33, wherein said kill queue comprises:

a plurality of entries, for storing a plurality of values of said kill signal generated during a corresponding plurality of clock cycles.

39. (new) The apparatus of claim 38, wherein each of said plurality of kill queue entries comprises a load data input, coupled to receive said kill signal.

40. (new) The apparatus of claim 38, wherein each of said plurality of kill queue entries comprises a hold data input, coupled to receive a current value of said entry.

41. (new) The apparatus of claim 38, wherein each of said plurality of kill queue entries comprises a shift data input, coupled to receive one of said plurality of values of said kill signal from one of said plurality of entries above said each of said plurality of kill queue entries.

42. (new) The apparatus of claim 38, wherein the instruction queue comprises a plurality of entries for storing a plurality of instructions, wherein said plurality of kill queue entries store corresponding said kill signal values for said plurality of instructions stored in said plurality of instruction queue entries.

43. (new) The apparatus of claim 33, wherein the instruction comprises a variable length instruction.

44. (new) The apparatus of claim 43, wherein said variable length instruction comprises an x86 architecture instruction.

45. (new) The apparatus of claim 43, wherein the instruction is provided to the instruction queue during the first clock cycle by an instruction formatter, said instruction formatter determining a length of the instruction.

46. (new) The apparatus of claim 33, wherein the instruction is output from the bottom entry of the instruction queue during the second clock cycle to an instruction translator for translating the instruction into one or more microinstructions to be selectively executed by the microprocessor based on said valid signal.

47. (new) A method for killing an instruction in a microprocessor, the method comprising:

loading an instruction into a first queue during a first clock cycle;

generating a kill signal during a second clock cycle subsequent to said first clock cycle;

loading a value of said kill signal into a second queue during said second clock cycle;

determining whether said value in the second queue is true during a third clock cycle in which said instruction is output from a bottom entry of said first queue;

generating a load signal for indicating during said third clock cycle whether said instruction was loaded into a bottom entry of said first queue during said first clock cycle, wherein if said load signal is false, said third clock cycle is subsequent to said second clock cycle; and

foregoing execution of said instruction if said value is true.

48. (new) The method of claim 47, wherein said third clock cycle is a same clock cycle as said second clock cycle.

49. (new) The method of claim 47, wherein said third clock cycle is clock cycle subsequent to said second clock cycle.

50. (new) The method of claim 47, further comprising:

formatting said instruction prior to said loading said instruction into said first queue.

51. (new) The method of claim 47, further comprising:

determining whether said instruction is shifted down in said first queue after said loading said instruction into said first queue; and

shifting down said value of said kill signal in said second queue after said loading a value of said kill signal into a second queue, if said instruction is shifted down in said first queue.

52. (new) The method of claim 47, further comprising:

predicting said instruction is a taken branch instruction, prior to said loading said instruction into said first queue;

detecting a misprediction of said branch instruction; and

said generating said kill signal during said second clock cycle in response to said detecting said misprediction.

53. (new) The method of claim 52, wherein a branch target address cache of the microprocessor performs said predicting said instruction is a taken branch instruction.

54. (new) The method of claim 52, wherein said misprediction of said branch instruction comprises a misprediction of a length of said branch instruction.

55. (new) The method of claim 52, wherein said misprediction of said branch instruction comprises a misprediction of an address of said branch instruction.

56. (new) The method of claim 52, wherein said misprediction of said branch instruction comprises said branch instruction being a non-branch instruction.

57. (new) The method of claim 47, further comprising:

branching the microprocessor based on a prediction that a branch instruction is taken, wherein said instruction is sequential to said branch instruction; and

said generating said kill signal during said second clock cycle after said branching the microprocessor.

58. (new) The method of claim 47, wherein said instruction sequentially follows a branch instruction predicted taken, the method further comprising:

said generating said kill signal during said second clock cycle in response to detecting said branch instruction is predicted taken.

59. (new) A program embodied on a computer readable medium, comprising:

computer-readable program code for providing an apparatus for killing an instruction loaded into an instruction queue of a microprocessor during a first clock cycle and output from a bottom entry of the instruction queue during a second clock cycle subsequent to the first clock cycle, said program code comprising:

first program code for providing a kill signal, for conveying a value generated during a third clock cycle subsequent to the first clock cycle;

second program code for providing a kill queue, coupled to said kill signal, for loading said kill signal value generated during said third clock cycle, and for outputting said kill signal value during the second clock cycle;

third program code for providing a load signal, coupled to said kill queue, for indicating during the second clock cycle whether the

instruction was loaded into the bottom entry of the instruction queue during the first clock cycle, wherein if said load signal is false, the second clock cycle is subsequent to said third clock cycle; and

fourth program code for providing a valid signal, coupled to said kill queue, generated during the second clock cycle for indicating whether the instruction is to be executed by the microprocessor, wherein said valid signal is false if said kill signal value output by said kill queue during the second clock cycle is true.